http: //lochness/patp.html

Ocean Modeling and Visualization on Massively Parallel Computer

Yi Chao, P. Peggy Li, Ping Wang, Daniel S. Katz, and Benny N. Cheng

Jet Propulsion Laboratory, California Institute of Technology

Abstract

Climate modeling is one of the grand challenges of computational science, and ocean modeling plays an important role in both understanding the current climatic conditions and predicting future climate change. Three-dimensional time-dependent ocean models require a large amount of memory and processing time to be run for realistic simulations. Recent advances in computing hardware, particularly the massively parallel processing (MPP) technology, have dramatically affected the prospect of studying the ocean at eddy-resolving resolutions. In addition to using advanced hardware, designing and implementing a well-optimized parallel ocean code will significantly improve computational performance and reduce the total computation time to complete these studies. Here we report our recent experiences in using the most widely used ocean model for climate applications and improving its computational performance on MPP machines. A number of optimization techniques have been developed to decrease the time required to solve ocean modeling problems. The resulting ocean code is about 2.5 times faster than the original code on the Cray T3D. A volume rendering tool was also developed on the MPP machines with the goal of performing interactive visualizations of large time-varying 3-dimensional scientific data sets. The optimized ocean code has been used to construct a 1/6 degree ocean model which has been integrated for 30 simulation years on the 256-processor Cray T3D. Preliminary results have shown that the solutions are significantly improved compared to previous coarser resolution calculations. It is anticipated that these and future high-resolution ocean models will determine the minimal resolution required by ocean models beyond which a further increase in resolution will have little qualitative improvement on the large-scale simulation.

1. Introduction

Observations suggest that the ocean plays an important role in the Earth's climate system. One of the principal roles of the ocean in the global heat balance is its ability to store and release heat, which moderates seasonal extremes and leads to the contrast between ocean and continent temperatures. The ocean is the major driving force in modulating natural climate variabilities, such as the El Niño phenomenon. In today's global warming environment, the ocean plays an important role in controlling the onset of the atmospheric warming (due to the increase of Greenhouse gases induced by the burning of fossil fuels) through absorbing the excess heat trapped in the atmosphere. It is therefore necessary to develop a comprehensive understanding of the world ocean circulation and its role in climate change.

Because of the enormous resource required to deploy and sustain ocean observations, it is virtually impossible to continuously monitor the ocean on the basin-to global-scale. In situ (e.g., ship-based) observations provide only sparse measurements over the world ocean. Despite its global coverage, satellite remote-sensed observations only provide information on the ocean surface. Information below the ocean surface has to be obtained from 3-dimensional numerical ocean models constrained by in situ and satellite observations. Therefore, ocean modeling is an indispensable tool to monitor the current climatic conditions and to predict the future climate change.

However, numerical ocean models need to be thoroughly tested against the available observations to establish their validity in describing the real world. This often involves conducting a large number of numerical experiments with different model configurations and parameters, and seeking the "best fit" between the model and observations. One of the biggest deficiencies in the existing ocean models is their inability to resolve meso-scale eddies in the ocean (equivalent to the storms in the atmosphere), whose spatial scale is on the order of 1 degree (or 100 km). One of the challenges in ocean modeling is to determine the minimum resolution of the model beyond which a further increase of resolution has

little qualitative impact on the model simulation.

The last decade has seen tremendous progress in exploring the role of horizontal resolution up to 1/3 [1] and 1/4 [2] degree using the parallel-vector computing technology. Eddy-resolving calculations beyond 1/4 degree are extremely costly and have only been feasible in recent years using massively parallel processing (MPP) technology. A group at Los Alamos National Laboratory (LANL) has performed a 10-year integration of a 1/5 degree ocean model on the 512-processor (PE) Connection Machine 5 (CM5) [3]. Using the 256-PE Cray T3D, JPL has conducted a 30-year integration of a 1/6 degree ocean model [4]. It is apparent that more eddy-resolving calculations with a resolution higher than 1 /6 degree are needed. A one-year integration of the 1/6 degree ocean model takes about 100 hours on a 256-PE Cray T3D. Given the limited computing resources available to the ocean modeling community and the requirement to conduct multi-decade integrations at even higher resolutions, it is therefore necessary to optimize the existing ocean code and reduce the total computation time needed to complete these eddy-resolving calculations.

High resolution models will undoubtedly generate large volume of data. A snapshot of the 1/6 degree North Atlantic model contains about 600 MBytes of data, If one saves the ocean model output every 3 simulation days (typical scale to resolve the synoptic events in the ocean), a 5-year time series of this model corresponds to about 300 GBytes of data. With existing commercial visualization software on high-end graphic workstations (e.g., the Power Onyx from Silicon Graphics, Inc.), one can only analyze a very small fraction of a data set of this size, not mentioning the time delay for data transfer between the MPP machines and local workstations. It is therefore advantageous to develop a compatible visualization software on the MPP machines where the ocean modeling is being conducted.

In this chapter, we report our recent experiences in running the ocean model and developing the visualization tool on the Cray T3D. After a brief description of our numerical ocean model, we present a few examples demonstrating how we improved its computational performance. The development of a volume renderer on the Cray T3D will be then described and its computational performance will be assessed, Finally, the scientific results using the optimized ocean model and the newly developed visualization tool will be presented. Given the short life cycle of MPP machines, the portability of the optimized ocean model and the volume rendering software will be discussed.

2. **Model Description**

We have selected the most widely used ocean model as our base code. This ocean model is derived from the Parallel Ocean Program (POP) developed at LANL [3], which evolved from the Bryan-Cox 3-dimensional primitive equations ocean model [5,6], developed at NOAA Geophysical Fluid Dynamics Laboratory (GFDL), and later known as the Semtner and Chervin model [2] or the Modular Ocean Model (MOM) [7].

The ocean model used in the present study solves the 3-dimensional primitive equations using the finite difference method. The equations are separated into barotropic (the vertical mean) and baroclinic (departures from the vertical mean) components. The baroclinic component is 3-dimensional, and is solved using explicit leapfrog time stepping. It can be parallelized very well on massively parallel computers. The barotropic component is 2-dimensional, and solved implicitly. It differs from the original Bryan-Cox formulation in that it removes the rigid-lid approximation and treats the sea surface height as a prognostic variable (i.e., free-surface). The free-surface model is superior to the rigid-lid model because it provides more accurate solution to the governing equations. More importantly, the free-surface model tremendously reduces the global communication otherwise required by the rigid-lid model.

3. Computational Considerations

The original POP code was developed in FORTRAN 90 on a LANL CM5 [3]. This code was then ported to the Cray T3D using Shared Memory Access (SHMEM) routines [8]. Since the code on the T3D was still time-consuming when large problems were encountered, improving the code performance was essential. In order to significantly reduce wallclock time, the code was optimized using single PE

optimization techniques [9] and other strategies. The remainder of this section discusses these strategies, the corresponding improvement in performance of the POP code on the Cray T3D, and issues of portability.

3.1 Memory Optimization and Arithmetic Pipelines

The Cray T3D uses the DEC Alpha EV4 processor with a 151 MHz clock and is capable of151 MFLOP/s of peak performance. The cache is 8 KBytes, direct mapped, and has 32 byte cache lines, There are several ways to achieve good performance on the T3D. One is through effective use of cache; another is through effective use of pipelined arithmetic. The POP code uses many two dimensional arrays, which can be inefficient when frequent stride-one addressing is encountered. One improvement is to change to explicit one dimensional addressing. For example, the two dimensional array KMU(IMT,JMT) can be replaced by KMU(IMT*JMT). This type of change can increase performance, both by simplifying index calculations and by making the code easier for the compiler to optimize.

Another useful strategy is to unroll loop statements. The DEC EV4 processor has segmented functional units for floating point multiply and addition. Although a multiply or addition can be issued every clock period, the result is not ready for 6 clock periods. (The divide operation needs 61 clock periods as it is not a pipelined function.) Thus, in order to get top performance from FORTRAN code, the user must expose functional unit parallelism to the compiler. Because of these features of the T3D system, the POP code has been optimized by unrolling loop statements where this exposes functional unit parallelism. The number of divide operations also has been minimized by using real variables to store the values resulting from divide operation and moving the divide operation out of loop statements when it is independent of the loop index. After these techniques were applied, the code performed significantly faster,

3.2 Using Optimized Libraries

There are several optimized libraries available on the T3D and other MPP machines, such as the Basic Linear Algebra Subproblems (BLAS) libraries [10], These libraries have been optimized to give the best possible performance when the user applies them properly. There are many matrix and vector computations in the POP code which consume a fair portion of the total computation time. We have replaced them by calling BLAS routines. For example,

```
DO I=1, IMAX
X( I)= ALPHA* (Y(I) *Z(I))
ENDDO
```

was replaced with a call to the extended BLAS routine named SHAD:

```
CALL SHAD(IMAX, ALPHA, Y,1, Z,1, 0. 0, X,1)
```

This can improve the performance of this particular loop by a factor of 5, for large values of I MAX.

3.3 Eliminating IF and WHERE Statements by Using Mask Arrays

In the original POP code, many logical IF and WHERE statements were used in distinguishing ocean points from land points. These statements consumed substantial computation time and reduced pipelining of operations inside loops. The compiler often was not able to efficiently optimize these loops (especially using automatic loop unrolling), since within the IF and WHERE statements some of the computations were quite complex. These IF and WHERE statements were replaced with land/ocean masking arrays, which store the values 1 for ocean points and O for land points, and then use multiplies of these values. For example, the statements:

```
WHERE (KMU. GE. 1)
VUF = SUF
ELSE
```

```
VUF = 0.0
ENDI F
```

were replaced by:

```
CALL SHAD(IMT'JMT, l. 0, UMASK, l, SUF, l, 0. 0, VUE', l)
```

where UMASK, VUF, SUF, and KMU are all of size (l MT, JMT), and UMASK was defined after the geometry initialization by:

```
DO I=1, IMT*JMT
IF (KMU(I, l). GE.1) THEN
UMASK(I, l) = 1.0
ELSE
UMASK(I, l) = 0.0
ENDIF
ENDDO
```

This could be done because KMU was strictly a function of input geometry, and it resulted in a speed-up of 3.2, for IMT = JMT = 100. We have also applied some mathematical formulae to eliminate some IF/WHERE statements. For example, the statements

```
IF (MIX. EQ.0) THEN
BETA = ALPHA
ELSE
BETA = THETA
ENDI F
```

where MIX equals either O or 1, are replaced by

```
BETA= ALPHA* ( l-MIX) +THETA*MIX
```

which improves the effective use of pipelined arithmetic.

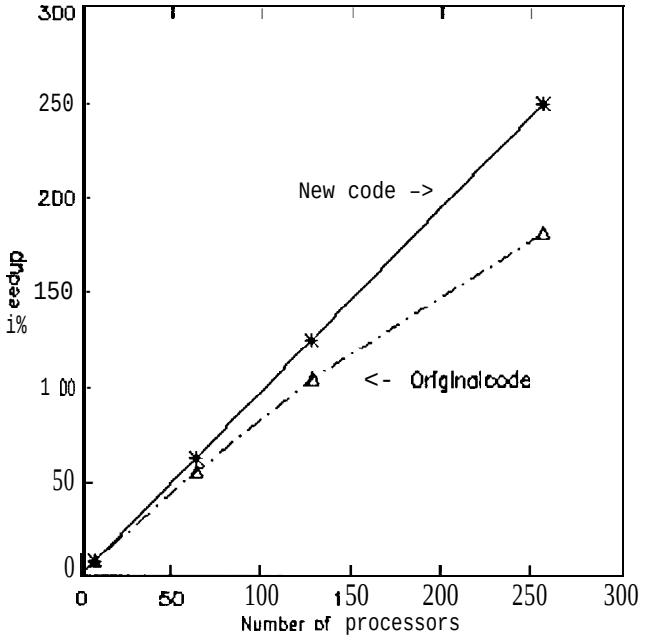
3.4 **Using Compiler Options and Parallel Software Tools**

In addition to the techniques previously discussed, the use of parallel software tools and compiler options were fully explored to maximize the performance of the ocean code. On the Cray T3D, the MPP Apprentice tool was used to monitor code performance to help us find and correct performance anomalies and inefficiencies. Performance measurements were taken for all PEs over the complete run. From this information, we could identify problems which negatively affected the performance and then apply the appropriate strategies. In addition to using manual optimization, automatic optimization was also applied through the compiler, such as the automatic unrolling option and the aggressive optimization option.

3.5 **Computational Performance**

Based on the above described optimization techniques, we have significantly improved the computational performance of the POP code. A test problem was chosen with a local grid size of 37 x 34 x 60 cells. Timings were run for machine sizes from 1 to 256 processors, corresponding to a global grid of up to 592 x 544 x 60. The POP code decomposes the grid in blocks in both x and y, and all z data for a given (x,y) is local to one processor. All results shown in this section refer to scaled size problems, where the problem size per processor is fixed and the number of processors is varied.

Figure 1 shows the run time (in wall clock time) per time step vs. the number of processors, for both the original code and the optimized code. The code running on one processor has been improved by 43°/0, and as the number of processors involved in the calculation increases, so does the improvement due to the optimization, up to 59°/0 on 256 processors.



It is clear from Figure 1 that the code’s scaling has also been improved. This improvement is enumerated in Figure 2, showing the speed-up achieved versus the number of processors used in the calculation, For the ideal parallel code, these two numbers would be equal. The optimized code is performing quite well, in running 250 times faster than the single processor code on 256 processors. The original code, however, clearly had scaling problems, as its speed-upon 256 processors is only 182 times.

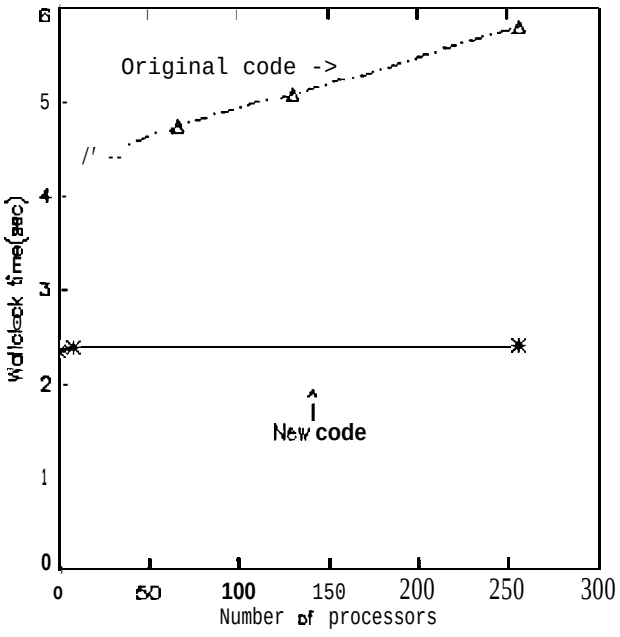
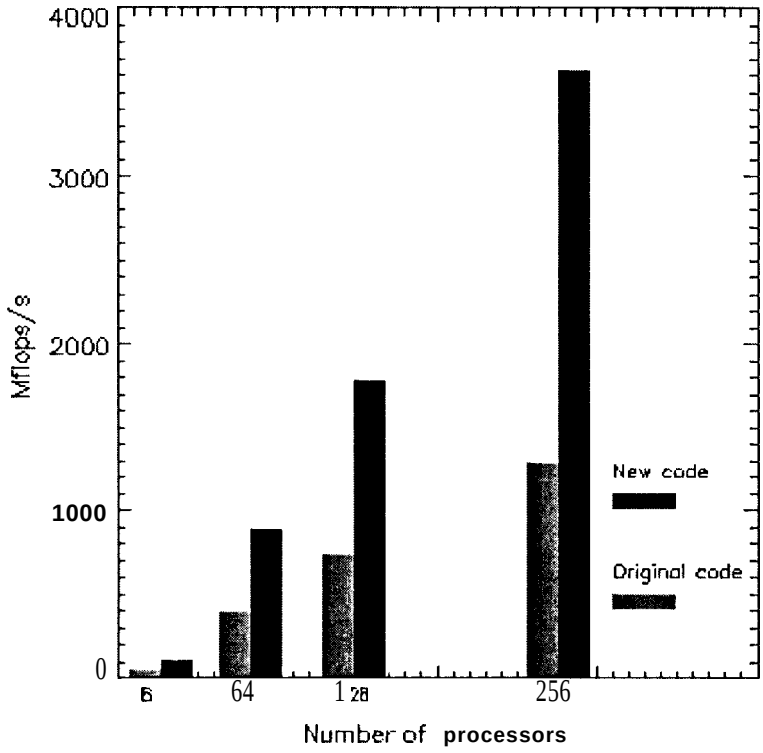


Figure 3 shows the actual performance achieved in terms of computation rate. As mentioned above, the T3D processor has a clock period of 6.6 nanoseconds, corresponding to a maximum floating point performance of 151.5 MFLOP/s. For 256 processors, this maximum performance is 38.8 GFLOP/s. The optimized code runs at over 3 GFLOP/s. It should be pointed out that the optimized code performs at

only 10°/0 of the peak machine speed, for several reasons. One is the ratio of computation to communication in these examples. If a larger local grid size was used, this ratio would increase and the overall performance would also increase, Another reason is poor cache reuse, due to the formulation of the code. It is written in terms of vector-vector routines (replaced by BLAS 1 [1 O] routines), rather than matrix-vector or matrix-matrix routines (which could be replaced by BLAS 2 [11] or BLAS 3 [12] routines). Neither of these make enough use of the data (each time it is loaded from memory) to achieve very high performance.



4. **Visualization on MPP Machines**

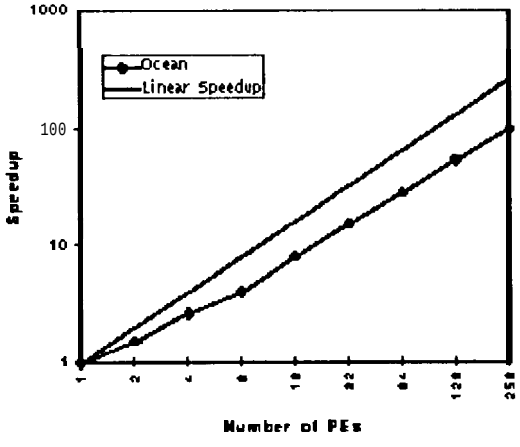
In this section, we present a new parallel volume rendering system, ParVox, for the visualization of 3-dimensional ocean data sets. The ocean model generates very large 3-dimensional data sets. For example, a 1/6 degree North Atlantic Ocean model has a dimension of 640x624x37 and there are five model variables, including temperature, salinity, and three velocity components. The data set at one time step is 590 Mbytes. If one saves the model output every three days, one year of simulation data is about 72 Gbytes. It becomes impractical and even impossible to visualize such large volume of data on existing high-end workstations. It is therefore necessary to use supercomputers to visualize and explore such a data set,

ParVox is designed to serve as either an interactive visualization tool for post-processing, or a rendering Application Programming Interface (API) to be linked with any application program. As a distributed visualization system, ParVox provides an X Window based GUI program for display and viewing control, a parallel input library for reading 4-D volume data sets in NetCDF format, a network interface program that interfaces with the GUI running on the remote workstation and a parallel wavelet image compression library capable of supporting both lossless and lossy compression. ParVox can visualize 3-dimensional volume data as a translucent volume with adjustable opacity for each different physical value, or as multiple isosurfaces at different thresholds and different opacities. It can also slice through the 3-dimensional volume and view only a set of slices in either of the three major orthogonal axes. Moreover, it is capable of animating multiple time-step 3-dimensional data sets at any selected viewpoint.

The detailed description of the ParVox system and its parallel implementation can be found in[13, 14]. In this section, a brief summary of the parallel splatting algorithm [15] is provided and how we use the one-sided communication functions in Cray's SHMEM library to optimize the efficiency of the parallel program is described. ParVox uses a combination of object-space decomposition and image-space decomposition. The input volumes are partitioned into small interleaving blocks distributed into each local processor's memory. First, each processor renders its volume blocks locally by splatting and compositing each voxel to the local processor's accumulation buffer. This results in a correct image for this portion of the volume. This sub-image is then composite with other sub-images from other processors. Communication is required to obtain the sub-image data from the appropriate processors. The rendered sub-images will overlap each other in depth as well as in the X and Y directions. Then, final compositing of the sub-images occurs and the distributed image (among all the processors) is reconstructed to a single image for final display.

As soon as a processor is finished rendering a block, it sends the individual final compositing regions off to the appropriate pre-assigned "compositing" processor, and then it proceeds to render the next block. In order to save memory and communication, we only send the valid pixels in an image region by putting a bounding box around the relevant data. The one-sided communication is accomplished using the Cray SHMEM library with the shmem_put call. However, with this approach, there is a complication if more than one processor needs to send data to a given compositing processor simultaneously. This is remedied by a semaphore mechanism. Each processor maintains a pointer to the memory buffer where the image region data will be put onto the remote compositing processor. The pointer is updated to point to the next available location by a remote processor using an atomic operation. The CRAY shmem_swap routine is used to implement this atomic operation. Once the pointer has been updated, the image region can be put to the remote processor's memory since there is no possible intrusion by any other processor. This one-sided communication (put) is very fast on the T3D (up to 120 MBytes/s). Since it involves no overhead on the other processor, communication can be easily overlapped with computation.

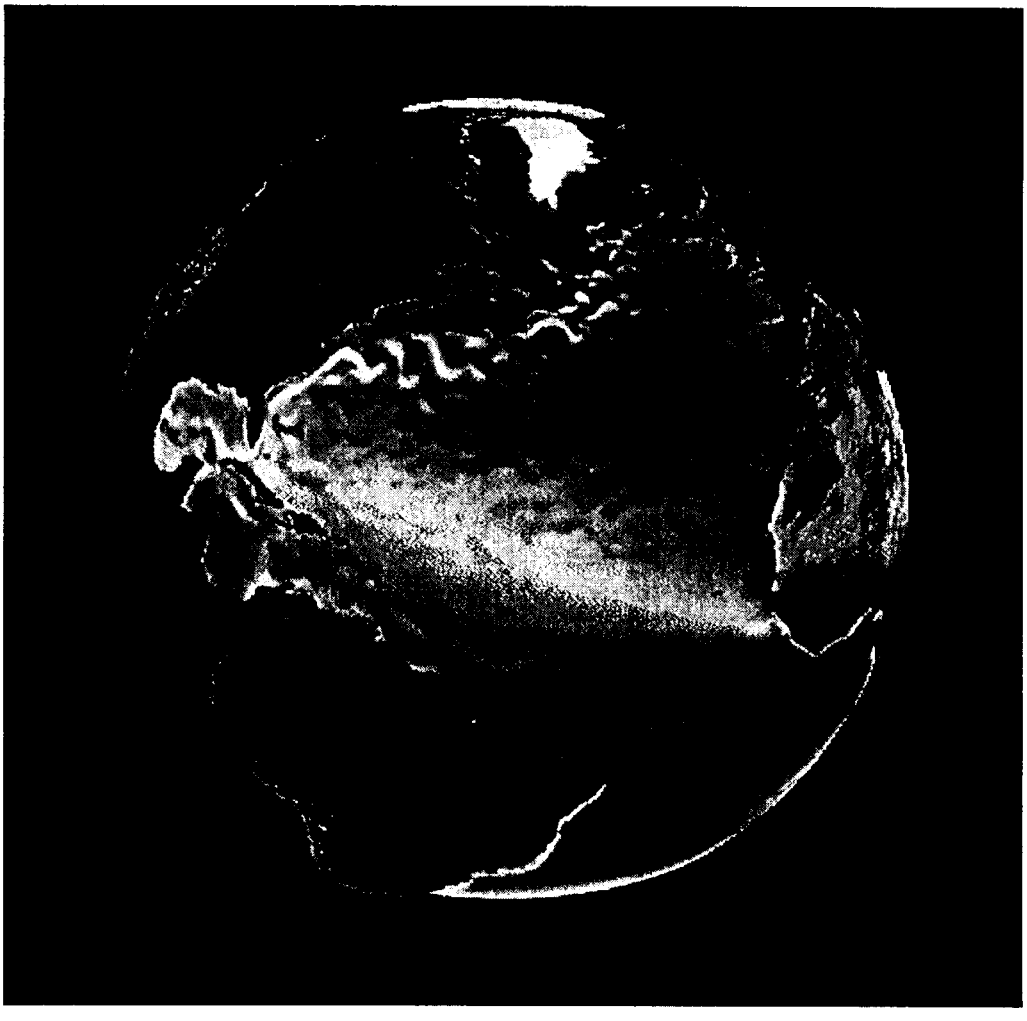
Figure 4 shows the speedup curve vs. number of PEs for the parallel splatting algorithm running on the Cray T3D. The input data is the 640x624x37 ocean temperature data set. In this benchmark, four blocks of input volume are assigned to each PE regardless of the total number of PEs. The image size is 512x5 12 and the final image is divided into square regions with 5 regions per PE. The rendering algorithm scales well from one PE to 256 PEs. More significant load imbalance is observed in the ocean data set with small number of processors due to the fact that the land voxels are grouped together in the data set and it is very likely that an entire block may contain no valid data.



5. **Scientific Results**

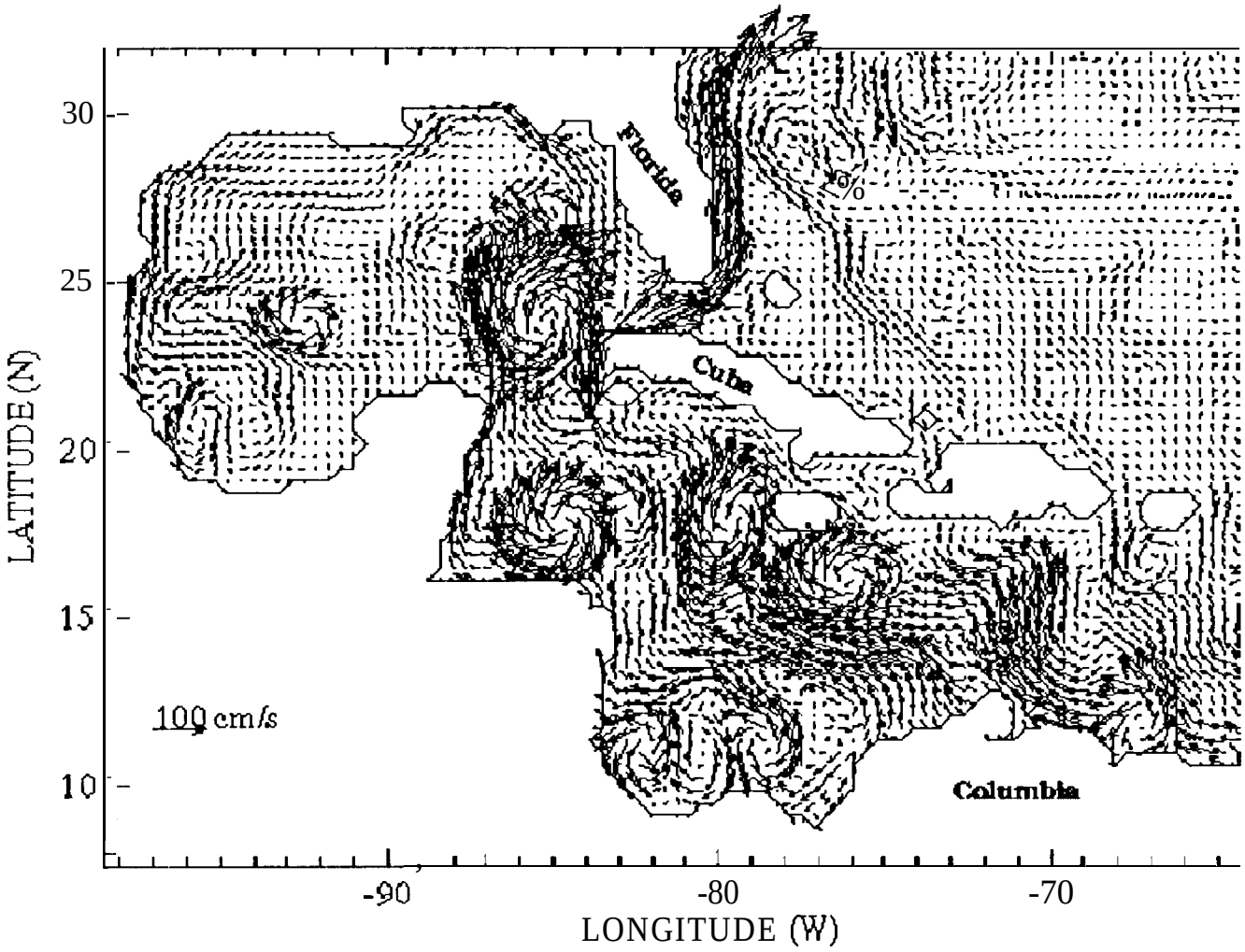
Using the optimized ocean model, we have performed a 30-year integration of a 1/6 degree ocean model on the 256-PE Cray T3D. Figure 5 shows a typical snapshot of the sea surface temperature overlaid by

the sea surface height. The ocean surface temperature is represented in color, while the sea surface height is used to created the shaded relief in the ocean as well as the elevation of the ocean surface [1 6]. In comparison with the previous eddy-resolving ocean model simulations [17, 18], this model shows improved Gulf Stream separation off the coast of Cape Hatteras [4]. As the horizontal resolution increases, increasingly fine scale features and the intensification of the currents are found with many of the larger scale features unchanged. It is quite promising that the physical processes responsible for both water mass and eddy formation can be reasonably simulated in models of this class.



In a regional application, we have documented the temporal and spatial evolution of meso-scale eddies in the Caribbean Sea and the Gulf of Mexico (Figure 6). These Caribbean eddies are quite regular, appearing about every 100 days. The eddies progress westward at a speed of near 15 ends, growing in amplitude. The above described 1/6 degree North Atlantic ocean model is able to reproduce major features of these Caribbean and Gulf of Mexico eddies, including their amplitudes, spatial and time scales, and propagation speed, comparing with satellite observations. Accurate description and understanding of these eddies are crucial for coastal monitoring and forecasting, which are of great benefit to the fishery and oil industries.

http: //lochness/patp.html



6. **Summary and Future Challenges**

We have developed a number of optimization techniques on the Cray T3D to improve the computational performance of the most widely used ocean model. These optimization techniques include unrolling loops, both manually and through the compiler, rewriting code to equivalence multi-dimensional arrays as one-dimensional arrays, simplifying algebra, reducing the number of divide operations, using mask arrays to trade fast floating point work for slow comparisons and branches, explicitly rewriting statements in array notation as loops, and using optimized libraries. All of these techniques contribute to the large decrease in time required to solve ocean modeling problems. The resulting ocean code is about 2.5 times faster than the original code on the Cray T3D.

We have also developed a parallel volume rendering system, ParVox, on the MPP system for interactive visualization of large time-varying 3-dimensional data sets generated by high-resolution 3-dimensional models. The ultimate goal of ParVox is to work in concert with the simulation models as a visualization, data exploration, and debugging tool for the scientists and model developers. A number of key features yet to be developed to fulfill such an objective. Among them, we are currently developing algorithms to support vector visualization, multiple clipping plans and slices, and function pipeline between the application and the visualization software.

We have conducted scientific studies using the optimized ocean model and the visualization software. A 1/6 degree ocean model was constructed on the 256-PE Cray T3D and was integrated for 30 simulation

years. Results have shown that the solutions are significantly improved compared to the previous 1/3 and 1/4 degree model simulations [17, 18]. Many of the physical processes responsible for both water mass and eddy formation can be reasonably simulated in models of this class. However, it still remains unknown what is the minimal resolution beyond which a further increase of resolution will have little qualitative improvement on the large-scale circulation. A group at LANL has just accomplished a 10-year integration of a 1/1 O degree ocean model on the 5 12-PE CM5, and their preliminary results show a further improvement than our 1/6 degree calculation [R. Smith, 1997, personal communication]. A short 2-year integration of a 1/12 degree ocean model has been attempted on the 5 12-PE Cray T3D [19]. With the upcoming Hewlett Packard (HP)/Convex SPP-2000, we are planning to construct a 1/16 degree ocean model. It is anticipated that these high-resolution ocean models will collectively reach some convergence on the minimal resolution required for climatic applications [20].

Given the short life cycle of the massively parallel computer, usually on the order of three to five years, we want to emphasize the portability of the ocean model, the associated optimization routines, and the visualization tool across several computing platforms. Thus far, our ocean modeling and visualization effort have primarily been conducted on the Cray T3D. We are now in the process of converting the ocean model from the Cray T3D to T3E. We have also started porting the ocean model to the newly available HP SPP-2000, using the MPI communication tools. Preliminary results show that our ocean model running on the SPP-2000 is about 3 to 4 times faster than on the T3D. It is expected that the performance can be further improved after implementing the above described optimization techniques. The current version of ParVox takes advantage of the asynchronous one-sided communication capability in the SHMEM library to overlap communication and computation. The newly released MPI 2.0 supports a new set of one-sided communication routines. ParVox can be readily ported to those MPP machines where the MPI 2.0 is supported.

References

1. F. Bryan, and W. Holland, A High Resolution Simulation of the Wind- and thermohaline-Driven Circulation in the North Atlantic Ocean, in Proceedings of Parameterization of Small-Scale Processes, Muller and Henderson Eds., Hawaii Institute of Geophysics Special Publication, 99-115, 1989.

2. A.J. Semtner and R.M. Chervin, Ocean-General Circulation from a Global Eddy-Resolviing Model, J. Geophys. Research Oceans, 97,5493-5550, 1992.

3. R.D. Smith, J.K. Dukowicz, and R.C. Malone, Parallel Ocean General Circulation Modeling, Physics D, 60,38-61, 1992.

4. Y. Chao, A. Gangopadhyay, F.O. Bryan, W.R. Holland, Modeling the Gulf Stream System: How Far From Reality?, Geophys. Res. Letts., 23,3155-3158, 1996.

5. K. Bryan, Numerical Method for the Study of the World Ocean Circulation, J. Comp. Phy., 4, 1687-1712, 1969.

6. M.D. Cox, Primitive Equation, 3-Dimensional Model of the Ocean, Group Tech. Report 1, GFDL/NOAA, Princeton, NJ, 1984.

7. R. Pacanowski, R.K. Dixon, and A. Rosati, Modular Ocean Model User's Guide, GFDL Ocean Group Tech. Report 2, GFDL/NOAA, Princeton, NJ, 1992.

8. SHMEM, Application Programmer's Library Reference Manual, publication SR-2165.

9. J.P. Brooks, Single PE Optimization Techniques for the CRAY T3D System, Benchmarking Group, Cray Research, Inc., 1995.

10. C, I.. Lawson, R.J. Hanson, D. Kincaid, and F.T. Krogh, Basic Linear Algebra Subprograms for FORTRAN Usage, ACM Trans. Math. Soft., 5, pp. 308-322, 1979.

11. J.J. Dongarra, J. Du Croz, S. Hammarling, and R.J. Hanson, Extended Set of FORTRAN Basic Linear Algebra Subprograms, ACM Trans. Math. Soft., 14, pp. 1-17, 1988.

12. J.J. Dongarra, J. Du Croz, 1.S. Duff, and S. Hammarling, A Set of Level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. SofL, 16 pp. 1-17, 1990.

13. S. Whitman, P. Li, and J. Tsiao, "Volume Rendering of Scientific Data on the T3D", Proceedings of the CUG 1996 Spring Conference, March 1996.

14. P. Li, et. al., "ParVox --A Parallel Splatting Volume Rendering System for Distributed Visualization", Proceedings of the 1997 IEEE Parallel Rendering Symposium, October 1997.

15. L. Westover, "Footprint Evaluation for Volume Rendering", Computer Graphics, Proceedings of Siggraph, Vol. 24, No. 4, pp. 367-376, 1990.

16. P. Li, W. Duquette, D. Curkendall, "RIVA --A Versatile Parallel Rendering System for Interactive Scientific Visualization", IEEE Transactions on Visualization and Computer Graphics, Vol.2, No. 3, pp.1 86-201, 1996.

17. A.J. Semtner, Modeling the Ocean General Circulation, Science, 269, 1379-1383, 1995.

18. J.C. McWilliams, Modeling the Ocean General Circulation, Annual Review of Fluid Mechanics, 28, 215-248, 1996.

19. R. Bleck, S. Dean, M. O'Keefe, and A. Sawdey, A Comparison of Data-Parallel and Message-Passing Versions of the Miami Isopycnic Coordinate Ocean Model (MICOM), Parallel Computing, 21, 1695-1720, 1995.

20. L.A. Drummond, J.D. Farrara, C.R. Mechoso, J.A. Spahr, Y. Chao, D.S. Katz, J.Z. Lou, and P. Wang, Parallel Optimization of An Earth System Model(100 GIGAFLOPS and Beyond?), 1997 Society for Computer Simulation (SCS) International Conference, Atlanta, Georgia, April, 1997.

6. Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under contract to the National Aeronautics and Space Administration (NASA). The Cray T3D used to produce the results in this paper were provided by the JPL Supercomputing Project. Fundings from the JPL Director Research and Development Fund (DRDF), NASA office of Mission to Planet Earth's Physical Oceanography program, and NASA High-Performance Computing and Communication (HPCC) program's Earth and Space Science Project (ESS) are greatly acknowledged. The authors wish to acknowledge Rick Smith, Phil Jones, and Bob Malone from LANL for providing the original POP code, David Shirley from Cray Research, Inc. for doing much of the work in the initial port of this code to the T3D, Alan Stagg from the Army CEWES for doing initial timing studies, Hong Ding from NERSC for initial optimization work, Martin Lewitt from HP for helping to implement the POP code on the SPP-2000, Edith Huang of JPL Supercomputing Project for helping to implement 10s on the Cray T3D, and Scott Whitman of Equator Inc. for the development of the parallel splatting algorithm and the network interface for ParVox, and James Tsiao of JPL for the development of the ParVox Graphic User Interface.

Figure Captions

Figure 1. Speed-Up (vs. single processor time) for scaled problems..

Figure 2. Wallclock time(sec) per time step for scaled problems.

Figure 3. Total floating point performance rates (MFLOP/s) for scaled problems,

Figure 4. The Speedup for the Parallel Rendering Algorithm

Figure 5. The sea surface temperature shaded relieved with sea surface height simulated by the 1/6 degree Atlantic Ocean model.

Figure 6. The surface current vector over the Caribbean Sea and the Gulf of Mexico simulated by the 1/6 degree Atlantic Ocean model.

Date: Mon, 3 Nov 1997 11:35:03 -0800
X- Sender-: llf@128 .149.33.200
Mime-Version: 1.0
1'0: Faye Elman <felman@mail1 .jpl .nasa.gov>
From: llf@paci fic.jpl.nasa.gov (Lee-Lueng Fu)
Subject: Re: Signature Authorization

I hereby give my approval of this request.

Lee-Lueng Fu

>Yi Chao has requested clearance for a book chapter "Ocean Modeling and
>Visualization on Massively Parallel Computer" with the URL
>http://lochness/patp .html . In order to process this, I
>would appreciate a written authorizat ion from you by ret urn email for- our
>records .
>
>Thank You.
>Faye Elman
>Document Review Services

Lee-Lueng Fu
MS 300-323
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Tel: 818-354-816"/
Fax: 818-393-6720
Email : llf@paci fic.jpl.nasa. gov

Date: Mon, 3 Nov 1997 11:35:03 -0800
X- Sender: llf@128 .149.33.200
Mime-Version: 1.0
To: Faye Elman <felman@mail1.jpl.nasa.gov>
From: llf@paci fic.jpl.nasa.gov (Lee-Lueng Fu)
Subject: Re: Signature Authorization

I hereby give my approval of this request

Lee-Lueng Fu

>Yi Chao has requested clearance for a book chapter "Ocean Modeling and
>Visualization on Massively Parallel Compu t er" with the URL
>ht tp:// lochne ss /patp.html. In order to process this, I
>would appreciate a written authorization from you by return email fOr our
>records.
>
>Thank You.
>Faye Elman
>Document Review Services

Lee-Lueng Fu
MS 300-323
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109

Tel: 818-354-816"/
Fax: 818-393-6 /20
Email: llf@paci fic.jpl.nasa.gov
